Memory Management

C# relies on the runtime to perform automatic memory management. The Com mon Language Runtime has a garbage collector that executes as part of your program, reclaiming memory for objects that are no longer referenced. This frees programmers from explicitly deallocating the memory for an object, eliminating the problem of incorrect pointers encountered in languages such as C++. C# does not eliminate pointers: it merely makes them unnecessary for most pro gramming tasks. For performance-critical hotspots and interoperability, pointers and explicit memory allocation is permitted in blocks that are marked unsafe.

مدیریت حافظه (Memory Management)  
C# برای مدیریت خودکار حافظه به runtime متکی است. Common Language Runtime (CLR) دارای یک garbage collector است که به‌عنوان بخشی از برنامه اجرا می‌شود و حافظه‌ی اشیایی را که دیگر ارجاعی به آن‌ها وجود ندارد، آزاد می‌کند. این ویژگی برنامه‌نویسان را از آزادسازی صریح حافظه‌ی اشیاء بی‌نیاز می‌کند و مشکل اشاره‌گرهای نادرست (dangling pointers) که در زبان‌هایی مانند C++ رخ می‌دهد را از بین می‌برد.

C# اشاره‌گرها (pointers) را حذف نکرده است؛ بلکه آن‌ها را برای بیشتر وظایف برنامه‌نویسی غیرضروری کرده است. برای بخش‌های حساس به کارایی (performance-critical hotspots) و یا هم‌کنش‌پذیری (interoperability)، استفاده از اشاره‌گرها و تخصیص صریح حافظه در بلوک‌هایی که به‌صورت unsafe علامت‌گذاری شده‌اند، مجاز است.

### توضیحات تکمیلی و نکات ارائه

1. **Garbage Collector (GC):**
   * به‌صورت خودکار حافظه اشیائی که دیگه استفاده نمیشن رو آزاد می‌کنه.
   * دیگه لازم نیست مثل C یا C++ دستی free() یا delete بزنی.
   * این موضوع باعث **کاهش خطاهای حافظه** میشه.
2. **مزیت اصلی:**
   * حذف مشکل بزرگ C++ → **Dangling Pointer** (وقتی حافظه آزاد میشه ولی همچنان بهش اشاره میشه).
   * کد تمیزتر و ایمن‌تر.
3. **Pointers در C#:**
   * حذف نشدن، فقط **لازم نیستن** در اکثر مواقع.
   * هنوز می‌تونی برای موارد خاص ازشون استفاده کنی (مثلاً بهینه‌سازی کارایی یا کار با APIهای سطح پایین).
   * این کار فقط در بلوک‌هایی که با کلیدواژه‌ی unsafe مشخص شدن مجازه.

 نکته: برای استفاده از unsafe باید تنظیمات پروژه اجازه بده (/unsafe compiler option).

 **کاربردهای واقعی pointers در C#:**

* **Interop:** وقتی می‌خوای با کدهای C/C++ یا APIهای سطح پایین کار کنی.
* **Performance:** برای عملیات سطح پایین مثل کار مستقیم با حافظه یا آرایه‌ها در شرایط خاص.

جمع‌بندی ارائه برای این قسمت:  
“در C# مدیریت حافظه به‌طور خودکار توسط Garbage Collector انجام میشه، بنابراین دیگه نیازی به آزادسازی دستی حافظه مثل C++ نداریم و مشکل اشاره‌گرهای نادرست هم حذف شده. با این حال، C# اشاره‌گرها رو کاملاً کنار نذاشته؛ در بلوک‌های unsafe همچنان میشه از اون‌ها برای کارایی بالا یا تعامل با زبان‌های دیگه استفاده کرد.”

Platform Support

C# has runtimes that support the following platforms: • • Windows 7+ Desktop (for rich-client, web, server, and command-line applications) • • macOS (for web and command-line applications—and rich-client applications via Mac Catalyst) • • Linux (for web and command-line applications) • • Android and iOS (for mobile applications) • • Windows 10 devices (Xbox, Surface Hub, and HoloLens) via UWP There is also a technology called Blazor that can compile C# to web assembly that runs in a browser.

### ترجمه پاراگراف

**پشتیبانی از پلتفرم‌ها (Platform Support)**  
C# دارای runtimeهایی است که از پلتفرم‌های زیر پشتیبانی می‌کنند:

* **ویندوز 7 به بعد (Windows 7+)** برای اپلیکیشن‌های دسکتاپ قدرتمند (rich-client)، وب، سرور و برنامه‌های خط فرمان.
* **macOS** برای اپلیکیشن‌های وب و خط فرمان — و همچنین اپلیکیشن‌های دسکتاپ (rich-client) از طریق **Mac Catalyst**.
* **Linux** برای اپلیکیشن‌های وب و خط فرمان.
* **Android و iOS** برای اپلیکیشن‌های موبایل.
* **دستگاه‌های ویندوز 10** (مثل Xbox، Surface Hub و HoloLens) از طریق **UWP (Universal Windows Platform)**.

همچنین فناوری‌ای به نام **Blazor** وجود دارد که می‌تواند کدهای C# را به **WebAssembly** کامپایل کند تا مستقیماً در مرورگر اجرا شوند.

توضیحات تکمیلی و نکات ارائه

1. چند سکویی (Cross-Platform):
   * C# محدود به ویندوز نیست؛ با .NET Core (و حالا .NET 8) روی macOS و Linux هم اجرا میشه.
   * همین ویژگی باعث شده C# انتخاب خوبی برای توسعه‌ی اپلیکیشن‌های سازمانی و مدرن باشه.
2. موبایل (Xamarin/.NET MAUI):
   * C# برای توسعه‌ی اپ موبایل هم استفاده میشه (Android و iOS).
   * الان با .NET MAUI میشه کد مشترک برای موبایل و دسکتاپ نوشت.
3. UWP و دستگاه‌های خاص:
   * برای توسعه‌ی اپ روی دستگاه‌های خاص مایکروسافت مثل Xbox یا HoloLens.
   * بیشتر در حوزه‌ی گیم و واقعیت ترکیبی کاربرد داره.
4. Blazor و WebAssembly:
   * تحولی بزرگ: میشه کد C# رو مستقیم داخل مرورگر اجرا کرد.
   * یعنی مثل جاوااسکریپت، ولی با C#.
   * این کار با WebAssembly انجام میشه که مثل یک ماشین مجازی سبک در مرورگر کار می‌کنه.
   * نتیجه: توسعه‌ی وب فقط با C# (بدون نیاز به جاوااسکریپت).

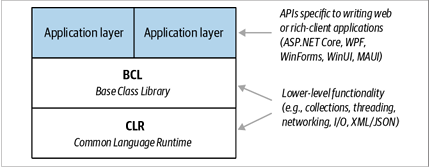
جمع‌بندی ارائه برای این بخش:  
“C# امروز یک زبان واقعاً چندسکویی شده؛ از ویندوز، مک و لینوکس گرفته تا موبایل (اندروید و iOS) و حتی دستگاه‌هایی مثل Xbox و HoloLens. علاوه بر اون، با Blazor میشه کدهای C# رو به WebAssembly تبدیل کرد و مستقیم در مرورگر اجرا کرد؛ یعنی دنیای وب هم برای C# باز شده.”

CLRs, BCLs, and Runtimes

Runtime support for C# programs consists of a Common Language Runtime and a Base Class Library. A runtime can also include a higher-level application layer that contains libraries for developing rich-client, mobile, or web applications (see Figure 1-1). Different runtimes exist to allow for different kinds of applications, as well as different platforms

### ترجمه پاراگراف

**CLR، BCL و Runtimeها**  
پشتیبانی زمان اجرا (runtime support) برای برنامه‌های C# شامل یک **Common Language Runtime (CLR)** و یک **Base Class Library (BCL)** است.  
یک runtime همچنین می‌تواند شامل یک لایه‌ی سطح بالاتر از اپلیکیشن باشد که کتابخانه‌هایی برای توسعه‌ی اپلیکیشن‌های دسکتاپ قدرتمند (rich-client)، موبایل یا وب را در بر می‌گیرد (شکل 1-1 را ببینید).  
Runtimeهای مختلف وجود دارند تا امکان ساخت انواع مختلفی از اپلیکیشن‌ها و همچنین پلتفرم‌های مختلف را فراهم کنند.



توضیحات تکمیلی و نکات ارائه

1. CLR (Common Language Runtime):
   * هسته‌ی اجرای کد C#.
   * وظایف اصلی: مدیریت حافظه، Garbage Collection، امنیت، JIT compilation (تبدیل IL به کد ماشین).
   * میشه گفت نقش ماشین مجازی (virtual machine) رو برای C# ایفا می‌کنه.
2. BCL (Base Class Library):
   * مجموعه‌ای از کلاس‌ها و توابع آماده برای کارهای پایه:
     + کار با رشته‌ها (String)
     + کالکشن‌ها (List, Dictionary)
     + IO (خواندن/نوشتن فایل)
     + شبکه، امنیت، Threading و …
   * مثل جعبه ابزار اصلی هر برنامه‌نویس C#.
3. Higher-Level Application Libraries:
   * بالای CLR و BCL، کتابخانه‌های تخصصی وجود دارن:
     + برای Rich-client apps → Windows Forms, WPF, MAUI
     + برای وب → ASP.NET Core
     + برای موبایل → Xamarin, MAUI
   * این لایه بهت امکان میده متناسب با نوع اپلیکیشن، ابزار مناسب رو انتخاب کنی.
4. چرا Runtimeهای مختلف؟
   * چون نیازهای اپلیکیشن‌ها و پلتفرم‌ها متفاوتن.
   * مثلاً:
     + .NET 8 runtime برای برنامه‌های cross-platform
     + Unity runtime برای بازی‌ها
     + UWP runtime برای اپ‌های ویندوز 10 دستگاه‌های خاص

جمع‌بندی برای ارائه:  
“هر برنامه‌ی C# روی یک runtime اجرا میشه. این runtime شامل CLR و BCL هست؛ یعنی ماشین مجازی و کتابخانه‌ی پایه. روی اون هم لایه‌های سطح بالاتر مثل ASP.NET برای وب یا MAUI برای موبایل قرار می‌گیرن. به همین دلیل runtimeهای مختلفی وجود دارن که متناسب با نوع اپلیکیشن و پلتفرم طراحی شدن.”

